Journal of Korea Multimedia Society Vol. 14, No. 10, October 2011(pp. 1275-1286)

A Systolic Array for High-Speed Computing of Full Search Block Matching Algorithm

Soon Ho Jung[†], Chong Ho Woo[††]

ABSTRACT

This paper proposes a high speed systolic array architecture for full search block matching algorithm (FBMA). The pixels of the search area for a reference block are input only one time to find the matched candidate block and reused to compute the sum of absolute difference (*SAD*) for the adjacent candidate blocks. Each row of designed 2-dimensional systolic array compares the reference block with the adjacent blocks of the same row in search area. The lower rows of the designed array get the pixels from the upper row and compute the *SAD* with reusing the overlapped pixels of the candidate blocks within same column of the search area. This designed array has no data broadcasting and global paths. The comparison with existing architectures shows that this array is superior in terms of throughput through it requires a little more hardware.

Key words: Video image processing, Motion compensation, Block matching algorithm, Systolic array.

1. INTRODUCTION

Applications such as video telephone, video-conference system, and high definition TV (HDTV), *etc.* require huge raw data for various digital image processing. A video compression algorithm (in the image processing such as MPEG) uses discrete cosine transform and motion compensation to remove redundancies [1,2]. We need motion estimation for the implementation of motion compensation. Although, many algorithms are proposed for it, FBMA is simple and widely used because it provides better image quality of images. The FBMA gives an optimal performance for blockwise motion estimation, but it requires a very massive computational overhead for real time processing [3-13]. Various VLSI array architectures are proposed in the literature to help solve this drawback. In these years some FPGA implementations are addressed [14,15].

Komarek and Pirsch[5] discussed the systolic array architecture for FBMA. This suffers from the low utilization and needs complex adders for accumulating the results in each column. Pan *et al.*'s architecture [7] uses two-stage BMA. Yea and Hu [8] consider a VLSI array architecture, which makes use of the overlapping feature in some parts of the search ranges to the adjacent reference blocks. Thus, this array shows the reduced number of input/output pins and an improvement in the computation speed. It requires broadcasting for propagating the input data to the processing elements and needs the global paths. Lee and Lu [9] use a semi-systolic array that contains two types of communication link, global connections for the search data and local connections for the partial sum. Lai *et al* [10,11] propose a 1-di-

※ Corresponding Author : Chong Ho Woo, Address : (608-810) Dept. of Computer Engineering, 599-1 Daeyoun-Dong Nam-Gu Pusan TEL : +82-51-629-6250, FAX : +82-51-629-6230, E-mail : chwoo@pknu.ac.kr
Receipt date : July 27, 2011, Revision date : Sep. 2, 2011
Approval date : Sep. 21, 2011
[†] Pukyong National University, Department of Computer Engineering
[††] Pukyong National University, Department of Computer Engineering
(E-mail: snow@pknu.ac.kr)
※ This research was supported by the Pukyong National University Research Abroad Fund in 2008(PS-2007-023)

mensional processing element array and two data-interlacing shift register arrays with data reuse. The current block pixels are sequentially inputted and broadcasted to all processing elements. Notion of data broadcasting and the global paths work against the premise of VLSI array as both limit the performance. Kittitornkun and Hu [12] propose a frame-level pipeline operation for the FBMA without using data broadcasting mechanism. But it requires long spiral interconnections and pipeline registers for delaying of input data. These high performance architectures can be used only under the condition of search range $p = N/2$ [8, 10-12].

This paper proposes a high-speed systolic array design for FBMA that does not require data broadcasting and global paths. We obtain the data dependence of FBMA for a reference block. The candidate blocks of search range in the algorithm are overlapped in row and column. The overlapped pixels are input once and are reused for adjacent candidate blocks. From the data dependence graph, we get a 2-dimensional VLSI array having time and space localities generated by the mapping methodology of systolic array. A row of the designed array computes the *SAD*s for the candidate blocks with reusing of the pixels within the same row of the search area. Each row of the array gets the search area pixels from the nodes in the upper row, so it can compute the *SAD*s with reusing of the pixels in the same column in the search area. As the pipeline period in the designed array is 2, we partition the processing elements of array into two groups and merge corresponding elements from each group to help improve the utilization of array processor. Assuming the reference block size of $N\times N$ and the maximum search range of p, the designed array takes $(N^2+2(p+1)N+6p)$ and $(3N+4p-1)$ clock cycles to obtain the motion vectors of the first and other reference block, respectively. There are $(N^2/2+1)(2p+1)$ processing elements with $(N+2p)$ input ports in the array. The designed 2-dimensional systolic array has many processing elements, but shows a higher computation speed as compared to the existing arrays.

2. FULL SEARCH BLOCK MATCHING ALGORITHM

The block matching algorithm estimates the amount of motion on a block basis between two successive frames as Fig. 1 [6]. The current frame is divided into a number of $N\times N$ blocks. This block is called as the reference block. Each *motion vector* (*MV*) is the displacement from the reference block in the current frame to a matching candidate block of search area in the previous frame. The search area in the previous frame is composed of the pixels located in p apart from the coordinates of the reference block, where p is the maximum search range.

The FBMA is the method that exhaustively searches for matching a reference block to the entire candidate blocks in the search area. A block is declared as the best matched block, if there exists a minimum difference between a reference and candidate blocks in the search area. The motion vector gives the difference of index between the selected candidate and reference blocks, and is as (1):

$$SAD(m,n) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|x(i,j) - y(i+m, j+n)\right|$$

$$MV = (m,n)_{\min(SAD(m,n))}, \quad -p \le m,n \le p \qquad (1)$$

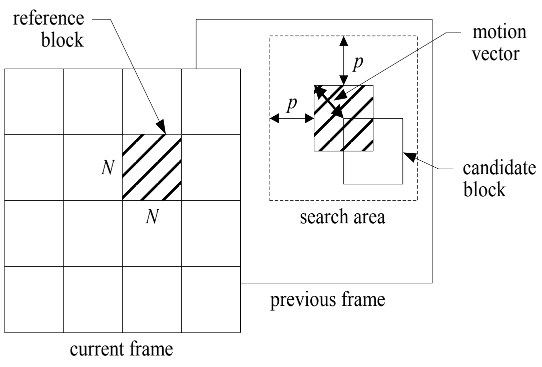


Fig. 1. The concept of full search block matching [6].

where $x(i, j)$ is the luminance value of the reference block, $y(i, j)$ is the luminance value of the candidate block, when the reference block size is $N\times N$, and the maximum search range is p. We use *sum of absolute difference (SAD)* as the matching criterion because it provides a simple measure.

The FBMA for a reference block is described as Fig. 2, where the reference block size is $N\times N$, and the maximum search range is p. The indices m and n designate the coordinates of the candidate block. We denote the indices m and n as 0 to $2p$ instead of $-p$ to p. The index k is a 1-dimensional index for the pixels in the reference block. So index k can denote $i\times N+j$, where indices i and j are coordinates of the pixel in the reference block as given in (1). The variable D_{min} is the minimum $SAD(m,n)$ for a reference block for which the index (m,n) is selected as the motion vector MV.

```
for ($m$=0 ; $m$<=2$p$ ; $m$++)
    for ($n$=0 ; $n$<=2$p$ ; $n$++)
        for ($k$=0 ; $k$<=$N\times N$-1 ; $k$++)
            $SAD(m,n) = SAD(m,n) + | x(k/N, k\%N) - y(m+k/N, n+k\%N) |$
        end for
        if ($D_{min} < SAD(m,n)$)
            $D_{min} = SAD(m,n)$
            $MV = (m,n)$
        end if
    end for
endfor
```

Fig. 2. The FBMA for a reference block.

3. DESIGN OF SYSTOLIC ARRAY

The systolic array features the properties of modularity, regularity, local interconnection, a high degree of pipelining, and highly synchronized multiprocessing. According to Kung's mapping methodology of the systolic array, first of all, a single assignment code is first derived from the sequential algorithm. And a dependence graph is derived, that is a graph that shows dependence of the computations that occur in an algorithm. By projecting this dependence graph and adapting the time-space localities, we get the systolic array. The projection direction is selected so that the internal organization of processing elements is simple and the number of input and output pins is small [3].

3.1 Single Assignment Code and Data Dependence Graph

The single assignment code is a kind of parallel representations of the algorithm. It assigns every data in the algorithm to only one variable. Thus, there is no dependence among variables in the code. From Fig. 2, we extract the overlapping data from the adjacent candidate blocks of the search area. These overlapping data are stored in the registers of the processing elements and shifted timely to the adjacent processing elements for the computations.

The dependence of the data in the search area for a reference block has relations as (2) and (3), where $y(m,n,k)$ means the k-th pixel in the candidate block with (m,n) coordinates. These mean that the adjacent candidate blocks have the same pixels. So the overlapped data within same row and column of blocks can be reused to compute the SADs for each candidate block.

$$y(m,n,k+\alpha) = y(m,n+\alpha,k) \tag{2}$$

$$y(m,n,k+\beta\cdot N) = y(m+\beta,n,k) \tag{3}$$

where $0 \le m,n \le 2p$, $0 \le \alpha,\beta < N$ and $0 \le k < N^2$

Adapting the dependent relations of (2) and (3) to algorithm in Fig. 2 and expanding the index space, we can derive the single assignment code for a reference block as given in Fig. 3. The minimum $SAD(m,n)$ is selected as D_{min} where m and n range between 0 and $2p$. In this case, (m, n) is the motion vector of the reference block as the final result.

From the single assignment code of Fig. 3, the data dependence graph for a reference block can

```
for ($m$=0 ; $m$<=2$p$ ; $m$++)
    for ($n$=0 ; $n$<=2$p$ ; $n$++)
        $SAD(m,n,0)$ = 0
        $D_{min}(m,n)$ = $\infty$
        for ($k$=0 ; $k$<=$N$*$N$-1 ; $k$++)
            if (($k$+1)%$N$==0 && $m$>0)
                $y(m,n,k)$ = $y(m-1,n,k+N)$
            else if(($k$+1)%$N$ != 0)
                $y(m,n,k)$ = $y(m,n-1,k+1)$
            end if
            if ($n$==0)
                $x(m,n,k)$ = $x(m-1,n,k)$
            else
                $x(m,n,k)$ = $x(m,n-1,k)$
            end if
            $SAD(m,n,k)$ = $SAD(m,n,k-1)$ + | $x(m,n,k)$ -
                $y(m,n,k)$ |
        end for
        $D_{min}(m,n)$=min($D_{min}(m,n-1)$, $SAD(m,n,N*N-1)$)
        $MV(m,n)$ = $(m,n)|_{Dmin(m,n)}$
    end for
    $D_{min}$ ($m$,2$p$)=min($D_{min}$ ($m$-1,2$p$), $SAD(m,2p,N*N-1)$)
    $MV(m,2p)$ = $(m,n)|_{Dmin(m,2p)}$
endfor
```

Fig. 3. The single assignment code of FBMA

be derived. There are 5 types of arcs (represented by the columns in $\vec{e}$) in the graph and are expressed as (4):

$$\vec{e} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & -N & 0 \end{bmatrix} \tag{4}$$

For example, when a block size N as 4 and the maximum search range p as 2, the 3-dimensional data dependence graph is shown in Fig. 4. An elliptic type node computes SAD and a square type node selects the candidate block with the minimum value of SAD and then provides MV.

A row composed of N^2 elliptic type nodes in Fig. 4 shows the computation of SAD for a reference block and a candidate block. A layer composed of ($2p$+1) rows of dependence graph represents the relation of the SADs computation for candidate blocks within same row in the search area. So the pixels within the same row can be reused to compute SADs for the adjacent candidate blocks. This dependence graph is composed of ($2p$+1) layers and each layer receives the overlapped pixel data of the search range from the upper layer and uses it for the operations of the next candidate block. It means that the pixels between rows are reused. So the pixels within the same row and column are reused for computing the SADs.

One layer of dependence graph is shown in Fig. 5. It requires (N^2+1) columns and ($2p$+1) rows for computing SAD and selecting the minimum value of SAD. First, input data of search area are modified as follows: Gray (circle type) nodes in the upper part of Fig. 5 are added. These virtual nodes offer no operations, but are used to reduce the input ports for the search area. Thus, the data of the search area are input to only every N-th nodes instead of inputting to all the nodes on the upper

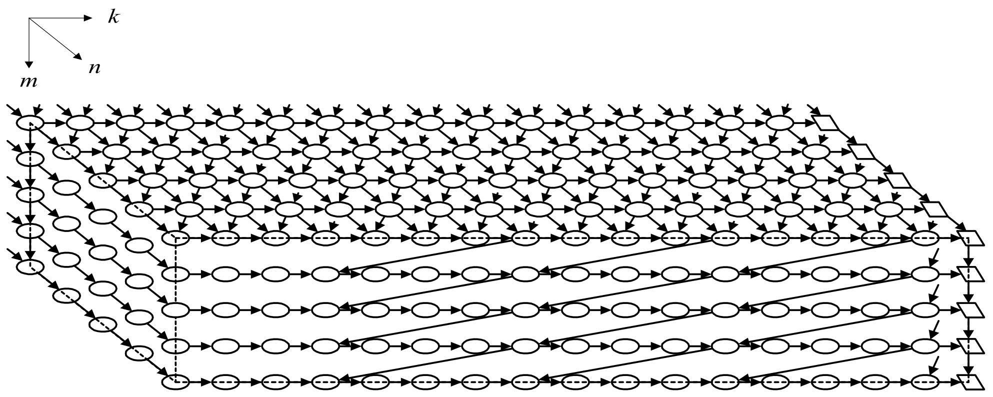


Fig. 4. Three dimensional dependence graph (N=4 and p=2).

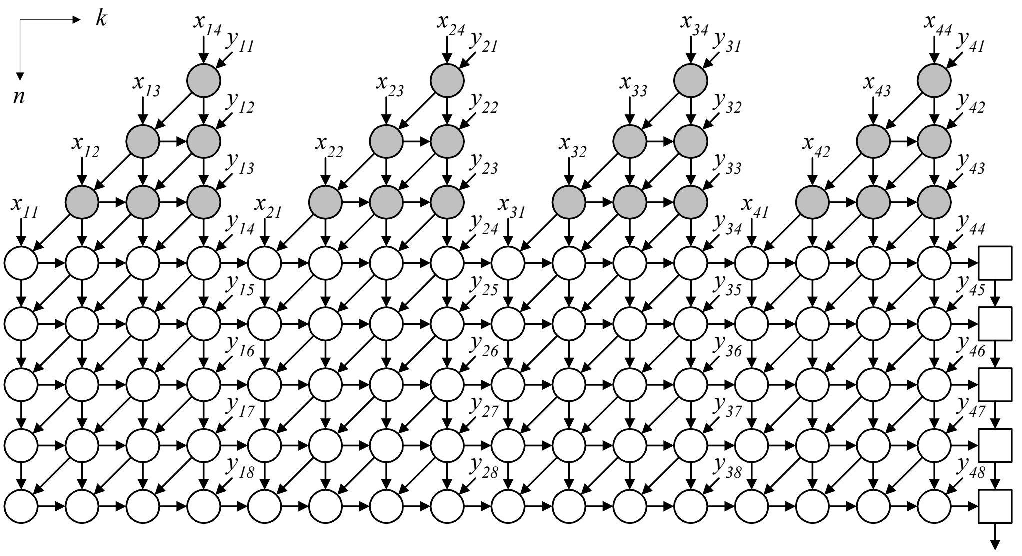


Fig. 5. A layer of the dependence graph (N=4, p=2).

boundary. Thus the number of input ports is reduced to N from N^2. The pixels of the search area are transferred in the [n k] = [1 −1] direction and the pixels within same row of blocks are reused. On the other hand, the data input of reference block is simpler than that of search area. For all candidate blocks, the same reference block is used. So pixels of the reference block are input to all the nodes on the upper boundary of the layer in the dependence graph and the input data are transferred in the [n k] = [1 0] direction. The computed *SAD*s are transferred in the [0 1] direction and the minimal *SAD* is selected in each layer. The selected *SAD* is transferred to the next layer and so we can select the best matched candidate block in the last layer. The every N-th node in Fig. 4 has the arc direction in [m n k] = [1 −N 0]. The pixels of the search area are transferred in this arc to next layer, and the overlapped data within the same column of the blocks are reused.

3.2 Deriving the Systolic Array

To derive the signal flow graph or systolic array from the dependence graph, we need the schedule vector $\vec{s}^T$ and projection vector $\vec{d}$. These vectors should satisfy the conditions of (5) and (6) for the maintaining the order of computations and the localities of time and space domains[3].

$$\vec{s}^T\vec{e} > 0 \tag{5}$$

$$\vec{s}^T\vec{d} > 0 \tag{6}$$

The selection of projection vector $\vec{d}$ is done as follows: The number of processing elements and input/output pins has to be smaller, and the internal organizations of processing elements have to be simpler in the signal flow graph or systolic array. To save silicon area and to enhance throughput rate, the projection vector $\vec{d}$ and the schedule vector $\vec{s}^T$ are heuristically chosen as (7) and (8).

$$\vec{s}^T = [N+1 \quad 2 \quad 1]^T \tag{7}$$

$$\vec{d} = [0 \quad 1 \quad 0] \tag{8}$$

Therefore, the pipeline period, α becomes 2 as given in Eq. (9). This means that a time unit of delay exists between data inputs of search area.

$$\alpha = \vec{s}^T\vec{d} = 2 \tag{9}$$

The resulting 2-dimensional array after projecting the dependence graph of Fig. 5 along the direction of (8) is depicted in Fig. 6. The N input

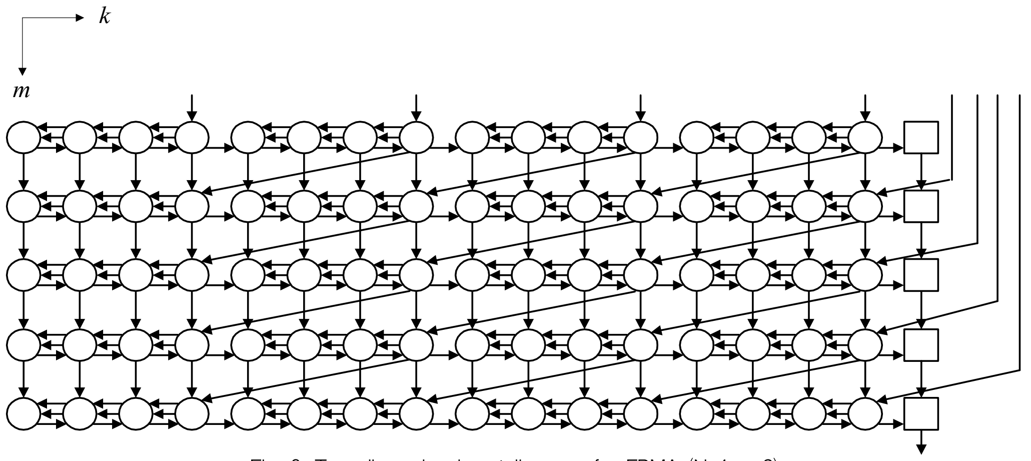


Fig. 6. Two dimensional systolic array for FBMA (N=4, p=2)

ports in the upper boundary are merged for input pixels of reference block and search area. So the data of the reference block is loaded in the processing elements while the data of search area is input and also transferred to the adjacent processing elements for the purpose of computing *SAD*. The minimal *SAD* is selected in each row. The minimal *SAD* and the moving vector (m, n) are obtained at the last row.

3.3 Localization of Data Path

Note, there exist the long crossing paths in the designed array of Fig. 6. We need to remove them for the high speed VLSI array. In order to remove the long paths of [1 $-N$] direction for transmitting the data of search area, we transform these paths to the local paths of [1 0] direction. The data of reference block are loaded in the processing elements and are not changed until they are used to complete the computations for all the candidate blocks in the search area. Thus, there is enough time to pass the data to the processing element of the lower row through the adjacent processing elements. The data of reference block loaded in the processing elements of each row are, then, passed to the adjacent processing elements in the [0 1] direction up to every N-th processing element and are turned down to the adjacent processing elements in the [1 1] direction. In this way, all paths in the array are localized in time and space domains. Of course the correctness of computations is maintained even if the array structure is slightly modified.

The designed systolic array architecture has some drawbacks. First, it has so many processing elements. Second, utilization of processing elements is low because the pipeline period is 2. To help alleviate this problem, we apply 'divide and rule' strategy. Note that only one of the adjacent processing elements within row is operated with the input data at a time. Hence, we can partition the processing elements of array into two groups and merge these two groups into one group. This not only reduces the number of the processing elements, but also improves their utilization. In this case, only one register for storing the input data of reference block is added to the processing elements. Consequently, the modified design is shown in Fig. 7. It does not alter computation time of the array.

3.4 Data Input Sequence

The designed array of Fig. 7 has input ports at the processing elements in upper row and right most processing elements for input of pixels in the reference block and the search area. Table 1 is de

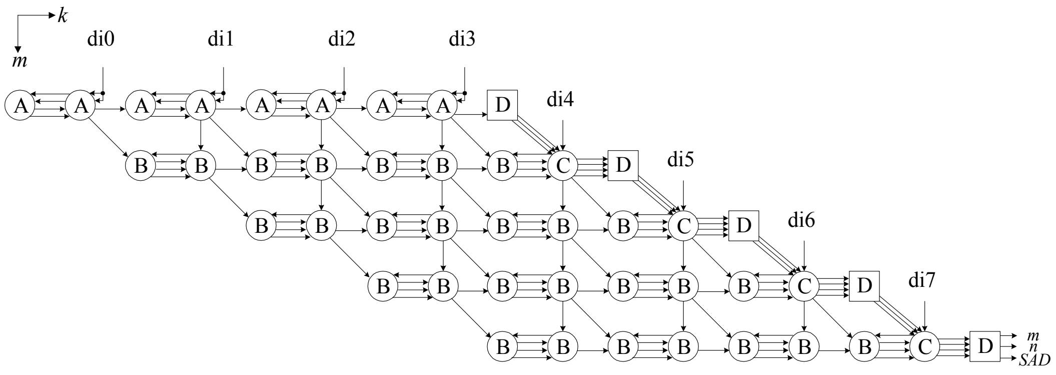


Fig. 7. The systolic array architecture for FBMA(N=4, p=2)

Table 1. Data input sequence in the designed array of Fig. 7 (*N*=4 and *p*=2)

input port / clock	di0	di1	di2	di3	di4	di5	di6	di7
1	x0							
2	x1							
3	x2							
4	x3							
5	y00	x4						
6	–	x5						
7	y01	x6						
8	–	x7						
9	y02	y10	x8					
10	–	–	x9					
11	y03	y11	x10					
12	–	–	x11					
13	y04	y12	y20	x12				
14	–	–	–	x13				
15	y05	y13	y21	x14				
16	–	–	–	x15				
17	y06	y14	y22	y30				
18	–	–	–	–				
19	y07	y15	y23	y31				
20	x'0	–	–	–				
21	x'1	y16	y24	y32				
22	x'2	–	–	–	y40			
23	x'3	y17	y25	y33	–			
24	y'00	x'4	–	–	y41			
25	–	x'5	y26	y34	–			
26	y'01	x'6	–	–	y42			
27	–	x'7	y27	y35	–	y50		
28	y'02	y10	x'8	–	y43	–		
29	–	–	x'9	y36	–	y51		
30	y'03	y11	x'10	–	y44	–		
31	–	–	x'11	y37	–	y52		
32	y'04	y12	y'20	x'12	y45	–	y60	
33	–	–	–	x'13	–	y53	–	
34	y'05	y13	y'21	x'14	y46	–	y61	
35	–	–	–	x'15	–	y54	–	
36	y'06	y14	y'22	y'30	y47	–	y62	
37	–	–	–	–	–	y55	–	y70
38	y'07	y'15	y'23	y'31	–	–	y63	–
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

picted the data input sequence, when a block size *N* is 4 and the maximum search range *p* is 2. The variables *x* and *y* means the pixel of the reference block and the search area, respectively. The indices of the variable *x* and *y* are the coordinates of the pixel. For example, the *x*11 and *y*11 means the pixel of (1,1) in the reference block and the search area, respectively. The variables x' and y' means the pixels of the next reference block and the corresponding search area, respectively.

Through the input ports (di0, di1, di2, di3 in Fig. 7) in upper row, the pixels of the reference block and the search area are input. First of all, the pixels of the reference block are input through these ports for *N* clock cycles and transferred in [0 -1] direction. These values are stored in the register of the processing elements. After storing the pixel of the reference block in the register of each *N*/2 processing element, these are transferred in [0 1] direction for transferring to lower rows of processing elements. At *N*/2 *time-th* processing elements, the pixels of reference block are transferred in [1 1] direction.

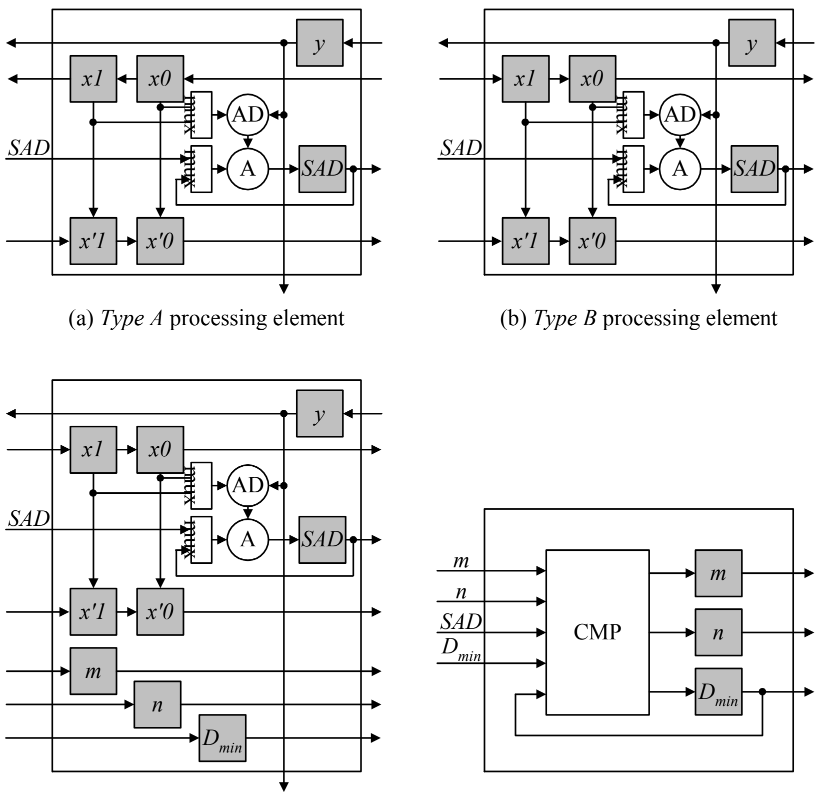
After input of the pixels in a reference block through the input ports in the upper boundary, (*N*+2*p*) pixels for the row of the search area are input through each port. These data are transferred in [0 -1] and [1 0] directions for data reusing in same row and column. The rightmost processing element of each row has a input port for pixels of the search area. The *SAD*s are computed between the pixels of stored reference block and transferred search area.

3.5 Internal Organizations of Processing Elements

As shown in Fig. 7, there are four types of processing elements denoted as *type A, type B, type C*, and *type D*. First three types compute *SAD*s with the input values of reference blocks and search area, while *type D* processing element selects the minimal *SAD*. In the following, we consider functions and design of these processing element types. The MUX, AD, A, and CMP in Fig. 8 stands for multiplexor, absolute difference, addition, and comparison operation respectively.

3.5.1 Type A processing elements

First row of the array in Fig. 7 has *Type A* processing elements. They receive the values of reference block and search area from the externals and compute *SAD*s. The data of the reference block and search area are transferred in [0 −1] direction, while the computed *SAD* is transferred in [0 1] direction. The gray box in Fig. 8(a) is a data register. Registers *x0* and *x1* store the reference block data because two adjacent processing elements are merged into one processing element. Register *y* stores data for search area and only one is needed to compute with two data of reference block. Registers *x'0* and *x'1* contain the intermediate values for transmitting of the reference block to the lower row. When the reference block size is $N\times N$ and the maximum search range is *p*, the processing element saves the inputted value of the reference block in registers *x0* and *x1* for first *N* clock cycles. After inputting the reference



(a) *Type A* processing element (b) *Type B* processing element

(c) *Type C* processing element (d) *Type D* processing element

Fig. 8. The internal organizations of the processing elements.

block, it saves the input data of $(2p+N)$ search area in register y for every two clocks. The values saved in registers $x0$ and $x1$ are transmitted to registers $x'0$ and $x'1$ at $(N+1)$-*th* clock. And then, the values of registers $x'1$ and $x'0$ are transmitted to [0 1] direction for the N clocks.

The absolute differences between the inputted reference block and the search area are accumulated at register *SAD*. The sum of input *SAD* and the absolute difference value between registers y and $x1$ is saved in register *SAD*. And the absolute difference value between registers y and $x0$ is accumulated at *SAD*. These two operations occur alternatively according to the state of the processing element.

3.5.2 Type B processing elements

Types B and *C* processing elements, shown in Fig. 8(b) and (c), respectively, are present in all other rows except the first row of Fig. 7. The data of the reference block in *type A* processing element are transferred in [0 -1] direction through registers $x'0$ and $x'1$. In case of *type B* and *type C* processing elements, the data are transferred in [0 1] direction through registers $x0$ and $x1$. The internal organization, however, remains unchanged except the transferring direction of the reference block.

3.5.3 Type C processing elements

The internal organization of the *type C* is similar to *type B* and contains registers D_{min}, m, and n. The *type C* processing element receives the minimal *SAD* and corresponding motion vector(m,n) from upper row, and transfers the data to *type D* processing element.

3.5.4 Type D processing elements

Type D processing elements are at the right end of array in Fig. 7. The unit CMP compares the inputs and selects the minimal *SAD* as shown in Fig. 8(d). The minimal *SAD* of each row is selected and stored in register D_{min}. The motion vector is stored in registers m and n. The D_{min} and motion vector selected in *type D* processing element are transferred through *type C* processing elements of the lower row.

4. RESULTS and DISCUSSION

Several factors determine the optimization criteria for the design of array processors. They include the number of processing elements, the processing time, and the number of I/O ports. The final choice of optimality criteria is dependent on the applications. A product of the array size (A) and the computation time (T) provides a useful measure for the hardware cost-effectiveness. In the application focused on the throughput, the AT^2 is used to compare the performance of VLSI array processors [3]. In the following, we consider the number of the processing elements, the computation time, and the number of input ports in the designed systolic array processor.

4.1 The number of Processing Elements

Designed systolic array of Fig. 7 has $(2p + 1)$ rows and each row has $(N^2/2 + 1)$ processing elements to compute *SAD* and to select the minimal *SAD*. Here the size of reference block is N and the maximum search range is p. Thus, total number of processing elements is:

$$\text{Number of Processing Elements} = (N^2/2 + 1)(2p + 1) \tag{10}$$

4.2 The Computational Time

The computational time T_A for a problem of the array process is derived from two nodes which have the maximum difference in the data dependence graph in computational time step as (11)[3]:

$$T_A = \max_{\vec{q},\vec{r}\in L}\left\{\vec{s}^T(\vec{q}-\vec{r})\right\}+1 \tag{11}$$

where L is the index set of the nodes in the data dependence graph. And the node $\vec{q}$ and $\vec{r}$ are the nodes that have the largest and the smallest time steps in computation of the problem. It means that the nodes are the first and the last computed nodes respectively through the selected schedule vector. The difference between two nodes, [$2p$ $2p$ N^2] and [0 ($-N$+1) (N−1)], is maximum in the data dependence graph of this design. So T_F of the designed VLSI array can be calculated by substituting these two index points and schedule vector of (7) to (11). The designed array process shares the input ports to get the data in reference block and search area. The initial time for preloading the reference block to the processing elements, N, is added. Hence, the computational time for the first reference block, T_F is computed as (12):

$$T_F = \begin{bmatrix} N+1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2p \\ 2p+(N-1) \\ N^2-(N-1) \end{bmatrix} + 1 + N \\ = N^2 + 2(p+1)N + 6p \qquad (12)$$

The time interval between the initiations of two successive problem instances in the process array is called as the block pipeline period. In the case that many problem instances are to be processed by the same array process, then the block pipeline period is used as a measure of processing speed in determining the optimization for the array processor [3]. If there are M problem instances to be computed, then the total processing time is $M\times T$, when the block pipeline period is T. After the data for the first reference block and the search area are input to the array, the data of the next reference block and the corresponding search area are input continuously without idle time as depicted in Table 1. For the initialization of the reference block, N unit times are needed. After inputting the reference block data, (N+$2p$) data for the search area are input. Thus, the data input of search area needs (2(N+$2p$)−1) unit times because the pipeline period α is 2. Hence, the block pipeline period T is ($3N$ + $4p$−1) as given in (13), and the motion vectors for the next reference blocks are computed at every T clock cycles.

$$T = N + 2(N+2p) - 1 \\ = 3N + 4p - 1 \qquad (13)$$

4.3 The number of input ports

In the proposed systolic VLSI array, the first row requires N input ports for the reference block and the search area, and while the remaining rows have one input port for the search area on the right side. Thus, the number of input ports is

$$\textit{Number of input ports} = N + 2p \qquad (14)$$

A comparison of the proposed architecture with existing architectures is illustrated in Table 2. According to CCIR Rec. 601, a video frame is 576× 720 and block size is 16× 16. Yea [8] and Lai [18]'s architectures show good performance, but they have global path and data broadcasting and use the

Table 2. Comparison of performances (Image size : 576× 720, block size : 16× 16, maximum search range : 8)

Architecture	Clock cycles per Block(T)	Clock cycles per Frame	Num. of Proc. Elements(A)	Num. of Registers	Num. of Data Accesses	Num. of Input Ports
T. Komarek[5]	544	881,280	273	4,739	8,192	16
S. B. Pan[7]	300	486,000	289	1,120	1,280	17
H. G. Yea[8]	256	423,936	256	512	768	3
C. Y. Lee[9]	256	414,720	256	3,488	1,217	4
Y. K. Lai[11]	256	414,720	256	1,024	752	3
Kittitornkun[12]	256	415,009	289	1,753	768	3
Proposed VLSI Array	79	128,493	2,193	13,443	1,280	32

search range of $-8/+7$. Kittitornkun [12]'s architecture is satisfied with the characteristics of the VLSI systolic array. It offers fewer memory accesses per block and the number of the processing elements. But these architecture can adapt only the case of $p = N/2$, and has long spiral paths.

The proposed array has no global path and does not require broadcasting. Hence, it can operate with higher clock rate. Compared to the previously proposed FBMA architectures, this architecture achieves the highest throughput. The performance improvement is at least 18% using AT^2 parameter in comparison, where A and T stand for chip area and processing time respectively. If the image size is larger, our architecture offers higher performance. The proposed array needs the registers in the processing elements for the data storing and the pipelining. The number of the register per a processing element is similar to architecture's of the Kittitornkun [12]. The proposed array needs a memory access per a pixel for a reference block and the search area. The overlapped pixels within same row and column of the search area are reused for computing the *SAD*s for the adjacent candidate blocks. The memory accesses for a search area are once at every two clocks, because pipeline period is 2. It makes keep the balance memory access of each input port and computation in the processing elements.

The proposed systolic array has many processing elements and input ports. This architecture needs 256 input pins when the case for the block size 16×16 with the maximum search range of 8. It is acceptable at current VLSI technology. Considering the advances in VLSI technologies, this hardware complexity is not a critical problem and the throughput is considered as more important measure. The proposed array processor is the proper VLSI architecture for getting the motion compensation of the real time video signal processing system requiring extremely high throughput.

V. CONCLUSION

In this paper, a high-speed systolic array architecture for FBMA is described. It is suitable for the application that demand high throughput. The designed 2-dimensional array compares concurrently a reference block with multiple candidate blocks by pipelined operations. A row of the designed array computes the *SAD*s with reusing of pixels within same row in the search area. Each row of array gets the overlapped pixels within same column in search area from the nodes of the upper rows. So the designed architecture reuses the pixels not only within the same row but also within same column in the search area. Because the designed array has only local path and no broadcasting of data, it can be operated in higher clock rate than other existing architectures with long paths or data broadcasting.

In case of the reference block size of $N\times N$ and the maximum search range of p, the motion vectors for a reference block are computed at every $(3N+4p-1)$ clock cycles. The proposed systolic VLSI architecture has $(N^2/2+1)(2p+1)$ processing elements and $(N+2p)$ input ports. We compared the designed VLSI architecture with the other existing architectures. While the proposed array uses more processing elements, its performance improvement in AT^2 criterion is better than that of existing architectures.

REFERENCES

[1] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. Legall, *MPEG video compression standard*, Chapman and Hall, 1996.

[2] S. Zhu and K. K. Ma, "A New Diamond Search Algorithm for Fast Block-Matching Mostion Estimation," *IEEE Trans. on Image Processing,* Vol.9, No.2, pp. 287-290, 2000.

[3] S. Y. Kung, *VLSI array processors*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[4] Y. Huang, C. Chen, C. Tsai, C. Shen, and L. Chen, "Survey on Block Matching Motion Estimation Algorithms and Architectures with New Results," *Journal of VLSI Signal Processing 42*, pp. 297-320, 2006.

[5] T. Komarek and P. Pirsch, "Array Architecture for Block Matching Algorithms," *IEEE Trans. on Circuits and Systems*, Vol.36, No.10, pp. 1301-1308, 1989.

[6] C. H. Hsieh and T. P. Lin, "VLSI Architecture for Block-Matching Motion Estimation Algorithm," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.2, No.2, pp. 169-175, 1992.

[7] S. B. Pan, S. S. Chae, and R. H. Park, "VLSI Architectures for Block Matching Algorithm," *IEEE Trans. Circuits and Systems for Video Technology,* Vol.6, pp. 67-73, 1995.

[8] H. G. Yea and Y. H. Hu, "A Novel Modular Systolic Array Architecture for Full-Search Block Matching Motion Estimation," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.5, No.5, pp. 407-416, 1995.

[9] C. Y. Lee and M. C. Lu "An Efficient VLSI Architecture for Full-Search Block Matching Algorithms," *Journal of VLSI Signal Processing,* Vol.15, pp. 275-282, 1997.

[10] Y. K Lai and L. G. Chen, "A Data-Interlacing Architecture with Two-Dimensional Data-Reuse for Full-Search Block Matching Algorithm," *IEEE Trans. Circuits and Systems for Video Technology,* Vol.8, pp. 124-127, 1998.

[11] Y. K. Lai, Y. L. Lai, Y. C. Liu, P. C. Wu, and L. G. Chen, "VLSI Implementation of the Motion Estimator with Two-Dimensional Data-Reuse," *IEEE Trans. on Consumer Electronics,* Vol.44, No.3, pp. 623-628, 1998.

[12] S. Kittitornkun and Y. H. Hu, "Frame-Level Pipelined Motion Estimation Array Processor," *IEEE Trans. Circuits and Systems for Video Technology,* Vol.11, No.2, pp. 248-251, 2001.

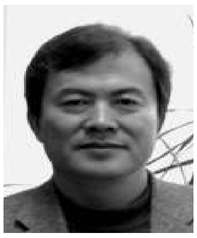
[13] S. Pandian, Hegde, J. Bala, and B. George, "A Study on Block Matching Algorithms for Motion Estimation," *International Journal on Computer Science and Engineering*, Vol.3, No.1, pp. 34-44, 2011.

[14] M. Azadfar, "Implementation of A Optimized Systolic Array Architecture for FSBMA using FPGA for Real-time Applications," *IJCSNS International Journal of Computer Science and Network Security*, Vol.8, No.3, pp. 46-51, 2008.

[15] G. Hegde, C. P. RajP, and P. R. Vaya, "Implementation of Systolic Array Architecture for Full Search Block Matching Algorithm on FPGA," *European Journal of Scientific Research ISSN 1450-216X*, Vol.33, No.4, pp. 606-616, 2009.

Soon Ho Jung

He received the B.S. degree in Mathematics Education from Seoul National University, Korea and the M.S. and Ph.D degree in Computer Science from Korea Advanced Institue of Science and Technology, Korea. He is currently a professor of Dept. of Computer Engineering in Pukyong National University, Korea. His teaching and reseach interests include Embedded Software, Ubiquitous Computing, HCI, Machine Learning.



Chong Ho Woo

He received his B.S., M.S., and Ph.D. degree from Kyungpook National University, Korea in 1978, 1981, and 1990, respectively. He is currently a professor of Dept. of Computer Engineering in Pukyong National University, Korea. His research interests include Embedded Systems, Sensor Networks, e-Learning Systems, and VLSI Array Processor for Image Processing